REMARKS

Claims 1-30 are pending in the present application. Claims 1, 11, and 21 are amended. Claims 1, 11, and 21 are amended to recite "wherein the superclass permission is a super class of the required permission." These features are supported at least on page 18, lines 5-32 and in Figure 5 of the current specification.

No new matter is added as a result of the above amendments. Reconsideration in view of the above amendment to claims in view of the following Remarks is respectfully requested.

I. Telephone Interview Summary

A telephone interview was conducted on June 27, 2005 with Examiner Abrishamkar with regard to features of independent claim 1. Applicants' representative submitted that Gong fails to teach determining if a superclass permission of a required permission is present in each protection domain of an access control context, wherein the superclass permission is a super class of the required permission. The Examiner indicated that the proposed amendments made to independent claim 1 potentially overcome the Gong reference, but further search and consideration is required before an agreement can be reached with the Applicants' representative.

II. Objection to Claim 9

The Office Action objects to claim 9 of the present invention because of informalities that the word "implied" is misspelled "implies." However, Applicants respectfully submit that the word "implies" is correct. On page 23, lines 23-28 of the current specification, an "implies" method is called on each permission of each PermissionCollection in the ProtectionDomains of the AccessControlContext. Thus, the name of the method is "implies", not "implied" as alleged by the Examiner. Accordingly, Applicants respectfully request the withdrawal of the objection to claim 9 of the present invention.

III. <u>Obviousness-type Double Patenting</u>

The Examiner has rejected claims 1-30 under the judicially-created doctrine of obviousness-type double patenting as being unpatentable in view of claims 1-30 of copending Application No. 10/002,488, which was filed November 1, 2001. Applicants have submitted herewith a terminal disclaimer under 37 CFR § 1.321, thus obviating the double patenting rejection. Applicants respectfully request that the rejection be withdrawn.

IV. <u>35 U.S.C. § 102(b), Alleged Anticipation, Claims 1-30</u>

The Office Action rejects claims 1-30 under 35 U.S.C. § 102(b) as being allegedly anticipated by Gong (U.S. Patent No. 10/002,439). This rejection is respectfully traversed.

Regarding independent claim 1, the Office Action states:

> In regard to claim 1, **Gong** discloses:
> -A method of controlling access to computer system resources based on permissions, comprising:
> -"***receiving a request for access to a computer system resource***" (Figure 7 item 750, column 6 lines 36-46, column 18 line 29-column 19 line 36);
> -"***determining if a superclass permission of a required permission is present in each protection domain of an access control context***" (column 6 lines 36-46, column 18 lines 29-45);
> -"**adding the required permission to a permission collection if the superclass permission of the required permission is present in each protection domain of the access control context**" (column 17 lines 1-5, column 19 lines 37-43); and
> -"***granting access to the resource if the superclass permission of the required permission is present in each protection domain of the access control context***" (column 10 lines 59-67, column 19 lines 4-36).

Office Action dated April 5, 2005, page 4.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that Gong does not teach every element of the claimed invention arranged as they are in claims 1, 11, and 21 of the present invention.

Amended independent claim 1, which is representative of amended claims 11 and 21 with regard to similarly recited subject matter, now recites:

> 1. A method of controlling access to computer system resources based on permissions, comprising:
> receiving a request for access to a computer system resource;
> <u>determining if a superclass permission of a required permission is present in each protection domain of an access control context, wherein the superclass permission is a super class of the required permission;</u>
> <u>adding the required permission to a permission collection if the superclass permission of the required permission is present in each protection domain of the access control context; and</u>
> <u>granting access to the resource if the superclass permission of the required permission is present in each protection domain of the access control context</u>.
> (Emphasis added).

Gong does not teach the features emphasized above. As discussed in the Abstract, Gong teaches a system for establishing complex security rules through the use of "permission" classes. For example, a permission superclass is established that defines an interface to a validation method. A permission subclass may be created which provides an implementation of the validation method indicating whether a given permission represented by one object belonging to a permission class encompasses the permission represented by another object belonging to a permission class.

However, Gong does not teach <u>determining if a superclass permission of a required permission is present in each protection domain of an access control context, wherein the superclass permission is a super class of the required permission</u>. The Office Action alleges that Gong teaches these features at column 6, lines 36-46, and column 18, lines 29-45, which reads as follows:

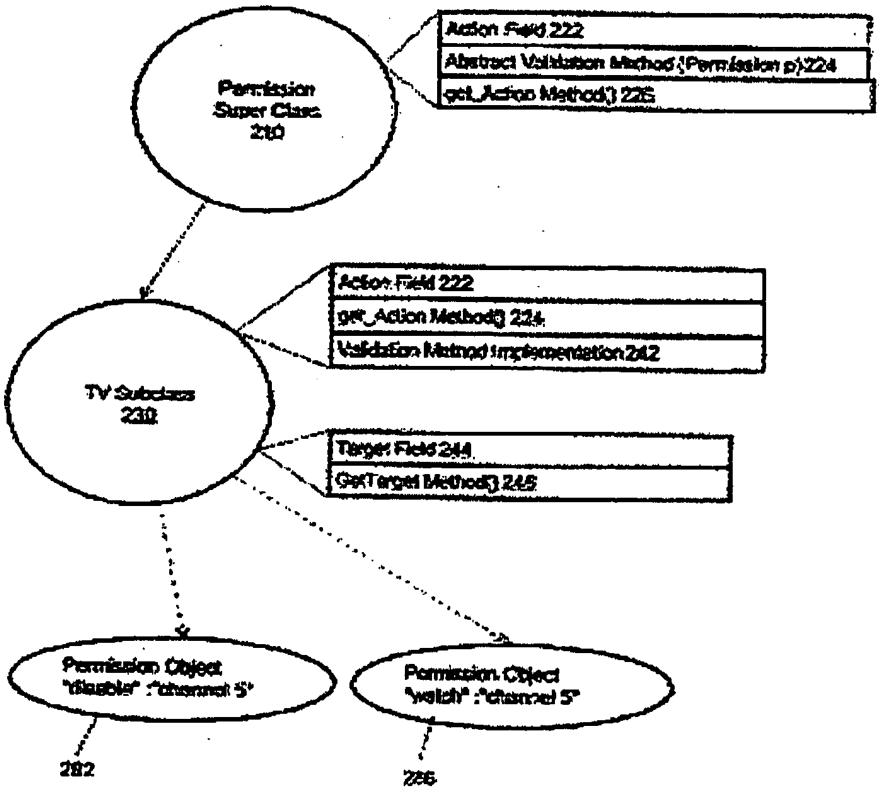
> When the validation method is invoked for a particular permission object belonging to a permission subclass, the validation method indicates whether a given permission is encompassed by the permission represented by the particular permission object. For example, the validation method of a permission object PO1 may be invoked to determine whether the

> permission represented by another permission object PO2 is encompassed in the permission represented by PO1, where both PO1 and PO2 belong to classes that descend from said permission super class.
>
> In step 754, a request is received for a determination of whether the action is authorized. The determination is based on the permission required to perform the action. In this example, the resource manager invokes an access controller to determine whether the permission required is authorized for the entity requesting access. The access controller receives the request and a required permission object which was transmitted by the resource manager.
>
> In step 754, the validation methods of one or more permission objects invoked in order to determine whether an action is authorized based on the permission required. An action is authorized if every protection domain object associated with an object requesting a determination of whether an action is authorized contains a permission represented by a permission object that encompasses the required permission for the action.

In the first section, Gong teaches invoking a validation method of a permission object PO1 to determine whether permission represented by another permission object PO2 is encompassed in the permission represented by PO1, where both PO1 and PO2 belong to classes that descend from the permission superclass. In the second section, Gong teaches authorizing an action if every protection domain associated with an object contains a permission represented by a permission object that encompasses the required permission for the action. At column 10, lines 59-67, Gong teaches that the abstract validation method includes code that ensures each attribute of permission represented by the object is implied by each attribute of the permission represented by the object whose validation method is invoked. For example, if the action field and target field of the permission object referred to by the object reference is <u>identical</u> to the action field and target field of the permission object whose validation method is invoked, the validation method returns a true value.

Thus, Gong teaches invoking a validation method of an object to determine if the permission attribute of the object is identical to the permission attribute of another object, where both objects belong to classes that descend from the permission superclass. Gong does not teach determining if a superclass permission of a required permission is present in each protection domain of an access control context. Rather, Gong merely teaches determining if the permission attribute of one object is identical to another object in every protection domain. In addition, Gong does not check to see if a superclass permission, which is a super class of the required permission, is present in each protection domain of an access control context. To the contrary, Gong teaches checking permission attributes of one object to see if those attributes are identical to another object, which also belongs to the classes that descend from the same permission superclass.

Gong's validation method is different from the presently claimed invention in that Gong's validation method operates on permission objects that are inherited from the same permission super class. **Figure 2** of Gong, which illustrates the permission objects, is shown below:



Permission Super Class 210
Action Field 222
Abstract Validation Method (Permission p) 224
get_Action Method() 226
TV Subclass 230
Action Field 222
get_Action Method() 224
Validation Method implementation 242
Target Field 244
GetTarget Method() 246
Permission Object "disable" : "channel 5"
Permission Object "watch" : "channel 5"
282
286

Fig. 2

As shown in **Figure 2**, permission object **282** and **286** are objects belonging to subclass TV subclass **230**, which is a subclass of permission super class **210** (column 9, lines 45-52). Validation method **242** in TV subclass **230** determines if the action field and target field of permission object **282** is identical to the action field and target field of permission object **286**. Thus, Gong's validation method operates on permission objects that are inherited from the same TV subclass **230**, which is a subclass of permission super class **210**. Gong's validation method does not determine if permission super class **210** is present in the required permission **282** or **286**. In fact, Gong is not concerned with whether permission super class **210** is present in permission object **282** or **286**. Gong is only concerned with permission attributes of permission objects **282** and **286**. Therefore, Gong does not and would not teach determining if a superclass permission of a required permission is present in each protection domain of an access control context, wherein the superclass permission is a super class of the required permission, as recited in claims 1, 11, and 21 of the present invention.

In addition, Gong does not teach <u>adding the required permission to a permission collection if the superclass permission of the required permission is present in each protection domain of the access control context</u>. The Office Action alleges that Gong teaches these features at column 17, lines 1-5 and column 19, lines 37-43, which reads as follows:

> If a protection domain is already associated with the code source, the domain mapper adds a mapping of the new class and protection domain to a mapping of classes and protection domains maintained by the domain mapper **448**.
>
> Typed permissions facilitate the establishment of new permissions. When a new category of permissions is desired, a new subclass is created. The particular rules or policy that govern whether the permissions granted a principal are encompassed by permission in the new category are implemented in the validation method of the new subclass representing permissions in the new subclass.

Gong teaches adding a mapping of the new class and protection domain to a mapping of classes and protection domains maintained by the domain mapper. In the second section, Gong teaches creating a new subclass when a new category of permissions is desired. However, Gong does not teach adding the required permissions to a permission collection if the superclass permission is present in each protection domain of an access control context. Rather, Gong teaches adding a mapping of new class and protection domain to a domain mapper and creating a new subclass for a new category of permissions. While Gong teaches, at column 12, lines 15-25, an add_permission method that adds a permission object to a set of permission objects contained in a permission collection object, Gong does not teach adding the permission object to the permission collection if the superclass permission is present in each protection domain of an access control context. To the contrary, Gong adds the permission object to the permission collection if the object is not of the same type (i.e. class) as any other permission object already contained in the PermissionCollection object. Thus, Gong adds the permission object only if there is no duplicate permission object in the PermissionCollection object, rather than if the superclass permission is present in each protection domain of the access control context. Therefore, Gong does not teach adding the required permission to a permission collection if the superclass permission of the required permission is present in each protection domain of the access control context, as recited in claims 1, 11, and 21 of the present invention.

Furthermore, Gong does not teach granting access to the resource if the superclass permission of the required permission is present in each protection domain of the access control context. The Office Action alleges that Gong teaches these features at column 10, lines 59-67, which is described above, and at column 19, lines 4-36, which reads as follows:

> In this example, the access controller first determines the protection domain associated with the first object represented on call stack **610**, which is object a. The protection domain associated with object a is protection domain I. The validation method of the first permission object, permission object **282** (in **FIG. 6**), is invoked, passing in the required permission object as a parameter. As mentioned earlier, the required permission represented by the required permission object is a permission to "disenable" "channel-5".
>
> When the validation method of the first permission object is invoked the validation method indicates that the required permission is not encompassed. Next, the validation method of permission object **286** (in **FIG. 6**) is invoked. The invocation of the validation method of permission object **286** indicates that the required permission is encompassed.

> The access controller then invokes the validation methods of the permission objects in the next protection domain object J, in the manner described. Each invocation of the validation methods of permission object **622** and permission object **626** indicates that the required permission is not encompassed.
>
> At step **764**, a determination is made of whether the action requested was authorized. If every associated protection domain contains a permission object that represents a permission encompassing the required permission, then the requested action is authorized. When the requested action is authorized control passes to step **768**, where the action is performed before execution of the steps ceases. In this example, because not every protection domain object contained a permission encompassing the required permission, performance of the steps ends. The requested action is not executed.

In the first section, Gong merely teaches a validation method that determines if each permission attribute of a permission object is implied by each attribute of the permission object represented by the object whose validation method is invoked. In the second section, Gong teaches invoking a validation method of first object in protection domain I that indicates the required permission is not encompassed. Then the validation method of the second object is invoked that indicates the required permission is encompassed. If every associated protection domain contains a permission object that represents a permission encompassing the required permission, the requested action is authorized. Thus, Gong teaches authorizing action if every protection domain has a permission object that encompasses the required permission. Gong does not grant access to the resource if the superclass permission of the required permission is present in each protection domain of the access control context. Therefore, Gong does not teach the features of claims 1, 11, and 21 of the present invention.

In view of the above, Applicants respectfully submit that Gong does not teach each and every feature of claims 1, 11 and 21. At least by virtue of their dependency on claims 1, 11 and 21 respectively, Gong does not teach the features of dependent claims 2-10, 12-20, 22-30. Accordingly, Applicants respectfully request the withdrawal of the rejection of claims 1-30 under 35 U.S.C. § 102(b).

In addition, Gong does not teach the specific features of claims 2-10, 12-20, and 22-30 of the present invention. For example, with regard to claims 5, 6, and 8, which are representative of claims 15, 16, 18, 25, 26, and 28 respectively with regard to similarly recited subject matter, Gong does not teach creating a new permission collection and adding the required permission to the new permission collection (claim 15), adding any subclass permissions of the required permission to the new permission collection (claim 16), or adding the permission to a permission collection associated with superclass permission (claim 18), if the superclass permission of the required permission is present in each protection domain of the access control context, when read in combination with independent claims 1, 11, and 21. As discussed above in arguments presented for claims 1, 11, and 21, Gong does not teach adding the required permission to a permission collection if the superclass permission of the required permission is present in each protection domain of the access control context, since Gong fails to teach determining if the superclass permission of the required permission is present in each protection domain of the access control context. Therefore, Gong does not and would not teach an adding step that comprises creating a new permission collection and adding the required permission to the new permission collection (claim 15), adding any subclass permissions of the required permission to the new permission collection (claim 16), or adding the permission to a permission collection associated with superclass permission, if the superclass permission of the required permission is present in each protection domain of the access control context, when read in combination with claims 1, 11, and 21. In addition, the Office Action alleges that Gong teaches these features at column 16, line 56 to column 17, line 13, which reads as follows:

> The domain mapper object **448** contains a mapping between classes and protection domains objects. Protection domain objects **482** contain a set or permissions. Protection domain objects are associated with the permission objects they contain, and with the classes to which a protection domain object is mapped to by domain mapper object **448**.
>
> Protection domain objects **482** are created when new classes are received by code executor **410**. When a new class is received, domain mapper **448** determines whether a protection domain is already associated with the code source. The domain mapper maintains data indicating which protection domains have been created and the code sources associated with the protection domains. If a protection domain is already associated with the code source, the domain mapper adds a mapping of the new class and protection domain to a mapping of classes and protection domains maintained by the domain mapper 448.
>
> If a protection domain object is not associated with the code source of the new class, a new protection domain object is created and populated

with permissions. The protection domain is populated with those permission that are mapped to the code source of the new class based on the mapping of code sources to permissions in the policy object. Finally, the domain mapper adds a mappling of new class and protection domain to the mapping of classes and protection domain as previously described.

In the above section, Gong merely teaches associating protection domain objects with permission objects they contain and with classes to which a protection domain object is mapped. The domain mapper creates protection domain objects when new classes are received and determines if the protection domain is already associated with a code source. If so, the domain mapper adds a mapping of the new class and the protection domain to a mapping of classes and protection domains maintained by the domain mapper. If a protection domain is not associated with the code source of the new class, a new protection domain is created and populated with permissions. The protection domain is populated with permissions that are mapped to the code source based on the mapping of code source to permission in the policy object.

Thus, Gong teaches adding a mapping of the new class and the protection domain to a mapping of classes and protection domains if a protection domain is associated with a code source or creating and populating a protection domain with permissions if the protection domain is not associated with the code source. Gong does not teach adding required permission or any subclass permissions to the new protection collection, or adding the permission to a permission collection associated with superclass permission, if the superclass permission of the required permission is present in each protection domain of the access control context, when read in combination with claims 1, 11, and 21.

To the contrary, Gong teaches adding permissions to a protection domain if the protection domain is not associated with the code source, rather than adding required permission, or any subclass permissions, to the new protection collection or adding the permission to a permission collection associated with the superclass permission, if the superclass permission of the required permission is present in each protection domain of the access control context. Therefore, Gong does not teach creating a new permission collection and adding the required permission to the new permission collection (claim 15), adding any subclass permissions of the required permission to the new permission collection (claim 16), or adding the permission to a permission collection associated with superclass permission, if the superclass permission of the required permission is present in each protection domain of the access control context, when read in combination with claims 1, 11, and 21.

With regard to claim 9, which is representative of claims 19 and 29 with regard to similarly recited subject matter, Gong does not teach that the determining step and the adding step are performed by a method called by the required permission in response to an implies method operating on the required permission. The Office Action alleges that Gong teaches these features at column 7, lines 30-45, which reads as follows:

> One permission can imply another. When one permission implies another permission, that one permission is said to encompass the other permission. For example, a permission to write to a directory, such "c:/", can imply a permission to write to any file in the directory, such as "c:/thisfile". Furthermore, an attribute of a permission can imply an attribute of another permission. For example, in some implementations, the action attribute of a permission to "write" implies an action attribute of a permission to "read".
>
> An amount attribute of a permission to withdraw three hundred dollars implies an other attribute of a permission to withdraw two hundred dollars.
>
> Usually, a permission encompasses another permission when all the permission attributes of one permission imply all the corresponding permission attribute of another permission.

In the above section, Gong teaches how an attribute of a permission can imply another attribute of another permission. But Gong does not teach a method that is called by the required permission performs the determining and adding steps. In fact, Gong does not even mention an implies method that operates on the required permission, let alone a method called by the required permission in response to an implies method that operates on the required permission. To the contrary, Gong only teaches a validation method in a permission object that determines if the permission encompasses permission of another permission object. Therefore, Gong does not teach a method that is called by the required permission in response to an implies method that operates on the required permission, as recited in claims 9, 19, and 29 of the present invention.

In view of the above, Applicants respectfully submit that Gong fails to teach the specific features of claims 2-10, 12-20, and 22-30 in addition their dependency on independent claim 1, 11, and 21. Accordingly, Applicants respectfully request the withdrawal of the rejection to claims 2-10, 12-20, and 22-30 under 35 U.S.C. § 102(b).

V. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: June 29, 2005

Respectfully submitted,

[illegible]

Duke W. Yee
Reg. No. 34,285
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

DWY/wym